Proyecto 1. Suma de procesos mediante Tuplas

201800679 – Pablo Fernando Victorio Morataya

Resumen

Este problema consiste en alojar objetos de bases de datos en sitios distribuidos, de manera que el costo total de la transmisión de datos para el procesamiento de todas las aplicaciones sea minimizado. Un objeto de base de datos es una entidad de una base de datos, esta entidad puede ser un atributo, un set de tuplas, una relación o un archivo. Los objetos de base de datos son unidades independientes que deben ser alojadas en los sitios de una red.

Una red de computadoras que consiste en un set de sitios donde un set de consultas es ejecutado, los objetos de base de datos requeridos por cada consulta, un esquema inicial de alojamiento de objetos de bases de datos, y las frecuencias de acceso de cada consulta desde cada sitio en un período de tiempo.

El problema consiste en obtener un nuevo esquema replicado de alojamiento que se adapte a un nuevo patrón de uso de la base de datos y minimice los costos de transmisión.

Palabras Clave

- Tublas
- Matriz de Frecuencia de accesos
- Árbol
- XML
- Matriz reducida de accesos

Abstract

This problem consists of hosting database objects in distributed sites, so that the total cost of data transmission for the processing of all applications is minimized. A database object is an entity in a database, this entity can be an attribute, a set of tuples, a relation or a file. Database objects are independent units that must be hosted by sites on a network.

A computer network consisting of a set of sites where a set of queries is executed, the database objects required by each query, an initial database object hosting scheme, and the access frequencies of each query from each site in a period of time.

The problem is to obtain a new replicated hosting scheme that adapts to a new database usage pattern and minimizes transmission costs.

Keywords

- Tuples
- Access Frequency Matrix
- Tree
- XML
- Reduced access matrix

Introducción

El problema de diseño de distribución consiste en determinar el alojamiento de datos de forma que los costos de acceso y comunicación son minimizados. Como muchos otros problemas reales, es un problema combinatorio NP-Hard.

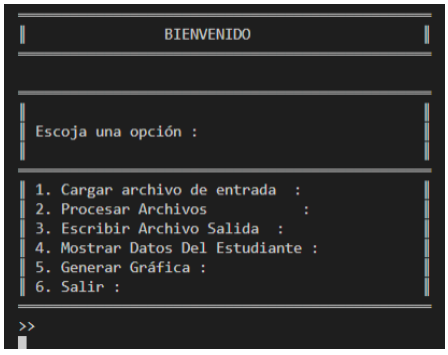
Algunas de las situaciones comunes que hemos observado cuando se resuelven instancias muy grandes de un problema NP-Hard son: Fuerte requerimiento de tiempo y fuerte demanda de recursos de memoria. Un método propuesto para resolver este tipo de problemas consiste en aplicar una metodología de agrupamiento.

Para “nt” tuplas y “ns” sitios, el método consiste en tener la matriz de frecuencia de acceso en los sitios F[nt][ns] de la instancia objetivo, transformarla en una matriz de patrones de acceso y agrupar las tuplas con el mismo patrón.

El patrón de acceso para una tupla es el vector binario indicando desde cuál sitio la tupla es accedida.

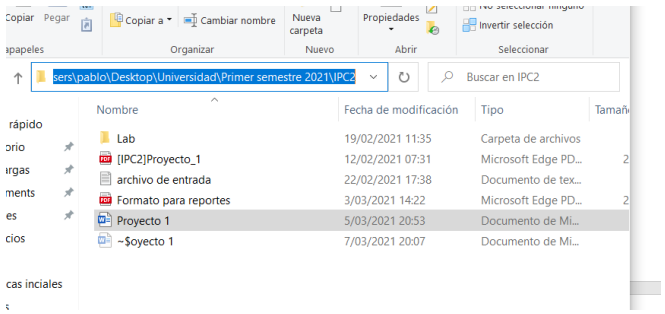
Desarrollo del Tema

Para el desarrollo del problema se subdividirá el mismo en 5 partes y un final, este mismo se explicará al final. Para poder acceder a cada una de estas opciones se definió un método menu() el cual funcionara como enlace con las demás clases.



- **Cargar archivo de entrada:**

En esta parte, se empezará a analizar la *matriz de frecuencia de accesos*, para esto, se le pedirá al usuario ingresar una dirección de entrada, obteniendo esta misma en la ruta del archivo



De esta forma se puede obtener la dirección para poder ingresarla en la consola del programa. El formato de la dirección tendrá una estructura similar al siguiente ejemplo:

“C:\Users\pablo\Desktop\Nuevacarpeta\entrada.xml “

Es importante mencionar que el formato de entrada será un archivo XML, por lo que de no cumplir con el formato de este tipo de archivo el programa mostrará un error.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<matrices>
    <matriz nombre="Ejemplo" n="5" m="4">
        <dato x="1" y="1">2</dato>
        <dato x="1" y="2">3</dato>
        <dato x="1" y="3">0</dato>
        <dato x="1" y="4">4</dato>
        <dato x="2" y="1">0</dato>
        <dato x="2" y="2">0</dato>
        <dato x="2" y="3">6</dato>
        <dato x="2" y="4">3</dato>
        <dato x="3" y="1">3</dato>
        <dato x="3" y="2">4</dato>
        <dato x="3" y="3">0</dato>
        <dato x="3" y="4">2</dato>
        <dato x="4" y="1">1</dato>
        <dato x="4" y="2">0</dato>
        <dato x="4" y="3">1</dato>
        <dato x="4" y="4">5</dato>
        <dato x="5" y="1">0</dato>
        <dato x="5" y="2">0</dato>
        <dato x="5" y="3">3</dato>
        <dato x="5" y="4">1</dato>
    </matriz>
</matrices>
```

Es importante resaltar que la etiqueta padre “matriz” tendrá identificadores *n* que corresponden a las filas de la matriz y *m* que

corresponden a las columnas de la matriz, si la cantidad de elementos dentro de la matriz no coincide con estos números antes descritos se mostrará un error en consola anunciando que el archivo de entrada posee algún error. Es importante mencionar que se pueden guardar la cantidad de matrices que desee, pues estas se guardan en una lista enlazada con un identificador único.

- **Procesar Archivos:**

En esta parte se procederá a realizar la conversión de *matriz de frecuencia de acceso* ***a*** una matriz binaria para poder comparar la misma y deducir los patrones de accesos que se tienen, cambiando cada uno de los accesos en su respectiva posición a valor de “1” independientemente de los accesos que tenga, o el valor de “0” si no tiene ningún acceso.

Para la siguiente matriz de frecuencia de acceso:

$$\begin{bmatrix} 2 & 3 & 0 & 4 \\ 0 & 0 & 6 & 3 \\ 3 & 4 & 0 & 2 \\ 1 & 0 & 1 & 5 \\ 0 & 0 & 3 & 1 \end{bmatrix}$$

Entonces, su correspondiente matriz de patrones de acceso es

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Con esta nueva matriz binaria se puede observar los patrones de acceso, por ejemplo, la fila 1 y la fila 3 tienen el mismo patrón de acceso, por lo que estas mismas se pueden relacionar entre si sumando sus valores de frecuencia, NO SUS VALORES BINARIOS.

Al utilizar el programa en consola esta opción dará al usuario la opción de escoger que matriz se desea procesar mediante su respectivo identificar en la lista.

```
╔═══════════════════════════════╗
║        PROCESO DE ARCHIVOS    ║
╚═══════════════════════════════╝

╟ Qué matriz desea procesar:    ╢
1

Calculando Matriz Binaria...
[1, 1, 0, 1]
[0, 0, 1, 1]
[1, 1, 0, 1]
[1, 0, 1, 1]
[0, 0, 1, 1]

Analizando posiciones de tuplas...

Sumando las tuplas...
Sumando las tuplas...
Suma final de tuplas...
[array([5, 7, 0, 6]), array([0, 0, 9, 4]), array([1, 0, 1, 5])]
```

Se podrá ver el proceso de nuestra matriz con la salida que el programa brindará por medio de la consola, mostrando la matriz binaria antes mostrada, y como analizará las posiciones de las *tuplas* para sumar en la *matriz de frecuencia de accesos.*

Luego se procederá a realizar la suma de las *tuplas*, al ser la posición 1 y posición 3 de la matriz binaria patrones repetidos, en la *matriz de frecuencia de accesos* se sumarán dichas posiciones, realizando todos los accesos posibles en un mismo procedimiento.

Por último, se mostrará en consola el resultado final de las sumas de *tuplas* realizadas, mostrando por “filas”, debido a que de esta forma se podrá observar de mejor manera los procesos realizados en la *matriz de frecuencia de accesos*, resultados finales y resultados sin procesos.

Para el ejemplo anterior de *matriz de frecuencia de accesos* se puede observar su matriz reducida:

$$\begin{bmatrix} 5 & 7 & 0 & 6 \\ 0 & 0 & 9 & 4 \\ 1 & 0 & 1 & 5 \end{bmatrix}$$

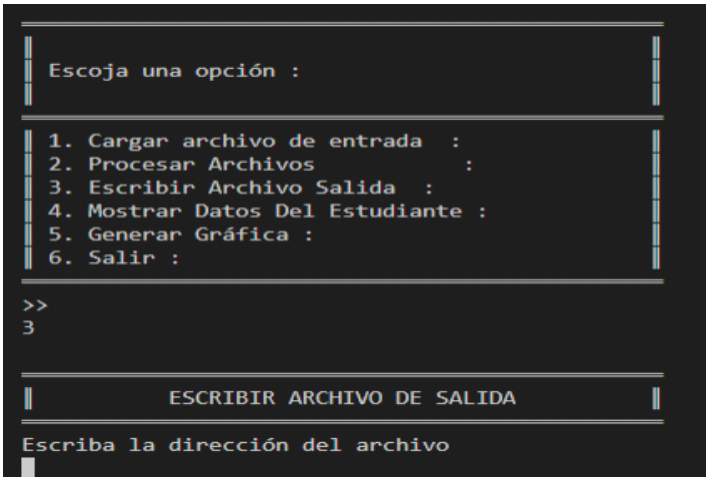
Una vez que el proceso haya terminado el método devolverá una matriz reducida como parámetro, para poder guardar esta misma en un archivo en formato *XML.*

Es importante aclarar que, si el archivo no es guardado y se realiza el proceso una vez más con una matriz diferente, la anterior quedará borrada pues este proceso es individual con el fin de no guardar archivos innecesarios y procesos repetidos, se guardará una matriz que el usuario este seguro de querer guardar.

- **Escribir Archivo Salida**

Este proceso esta ligado a la misma función de abrir un archivo *XML,* pues el usuario brindará una dirección para poder guardar el archivo de salida. Para poder ver como obtener una dirección mediante una carpeta regresar la primera opción del menu() de accesos.

Es importante aclarar que si no se ha procesado una *matriz de frecuencia de accesos* esta tendrá un objeto “Nulo” o “None”, en el framework Python, y hasta que se haya procesado una matriz el usuario podrá guardar un archivo de salida, esto con el fin de no guardar archivos blancos o vacíos por alguna equivocación al seleccionar esta opción de manera accidental.

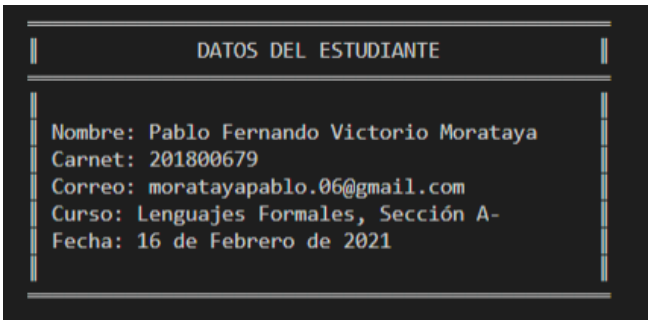


El formato a introducir dentro de la consola será:

“C:\Users\pablo\Desktop\Nuevacarpeta\Nombre del archivo de salida.xml”

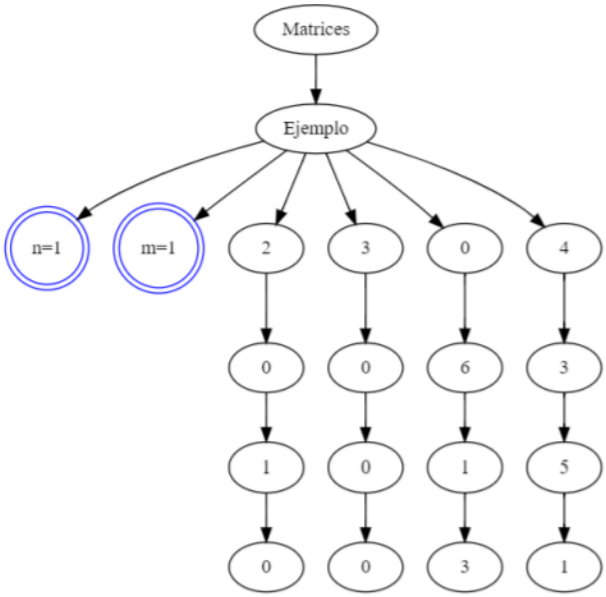
- **Mostrar datos del estudiante**

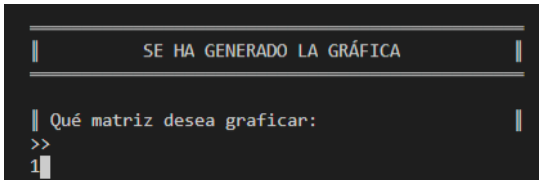
Un pequeño apartado poder conocer los datos del programador de dicho programa.



- **Generar Gráfica**

En esta parte se podrá seleccionar una *matriz de frecuencia de accesos* para poder graficar en una estructura de árbol, siguiendo la estructura de *árbol* del archivo *XML* siendo la raíz “Matrices” y “Matriz” el nodo padre del cual se van a subdividir cada uno de los nodos hijos



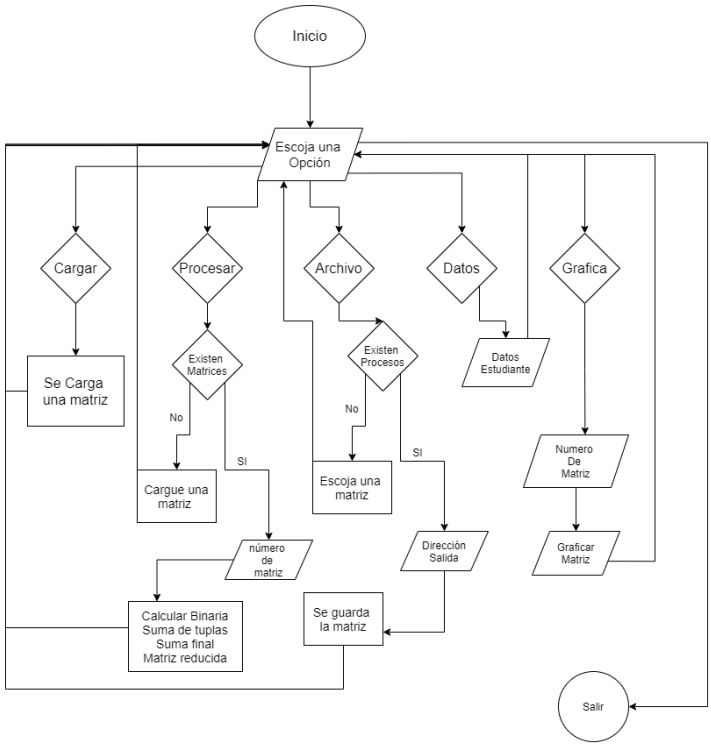


Se usará la librería graphviz para poder realizar la grafía que se mostró antes, esta librería permite realizar cada uno de los nodos, siendo los círculos de manera independiente, y nombrados cada uno de manera diferente, con su propio identificador, y señalando con sus respectivos apuntadores.

- **Salir**

El método menu() funciona con un ciclo while repitiendo el ciclo siempre que no se seleccione la opción salir, cuando esta se seleccione el estado pasará a un estado falso y el ciclo se detendrá.

Diagrama de Flujo



Conclusiones

Es importante detectar patrones de accesos cuando se realicen procesos. Tomando esa idea se puede ejemplificar con el navegador Chrome. Este realiza un proceso para cada acceso que necesite siendo algo muy poco optimo para la computadora del usuario.

Mediante la suma de tuplas se reconocieron patrones de accesos, al hacer esto, se pueden optimizar procesos, pues estos mismos se pueden aplicar a los accesos necesarios que tengan el mismo patrón.

La principal idea de realizar este tipo de estructura con un TDA, es familiarizar al programador o estudiante al manejo de una l manejo de una “pila”, pues para próximos proyectos es importante generar el hábito de realizar una excelente programación, optima y funcional.

Referencias bibliográficas

C. J. Date, (1991). An introduction to Data base Systems. Addison-Wesley Publishing Company, Inc.